Database Processing
CS 451 / 551

Lecture 9:

Query Processing

UNIVERSITY OF OREGON

Suyash Gupta

Assistant Professor

Distopia Labs and ONRG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

Assignment 2 is Out!
Deadline: Nov 13, 2025 at 11:59pm

Quiz 2: Nov 6, 2025 (in class)

What is meant by Query Processing

- Any activity that is related to extracting data from the database.

What is meant by Query Processing

- Any activity that is related to extracting data from the database.

- We can sub-divide query processing into three tasks:
    - Parsing and translation.
    - Optimization.
    - Evaluation.

What is meant by Query Processing

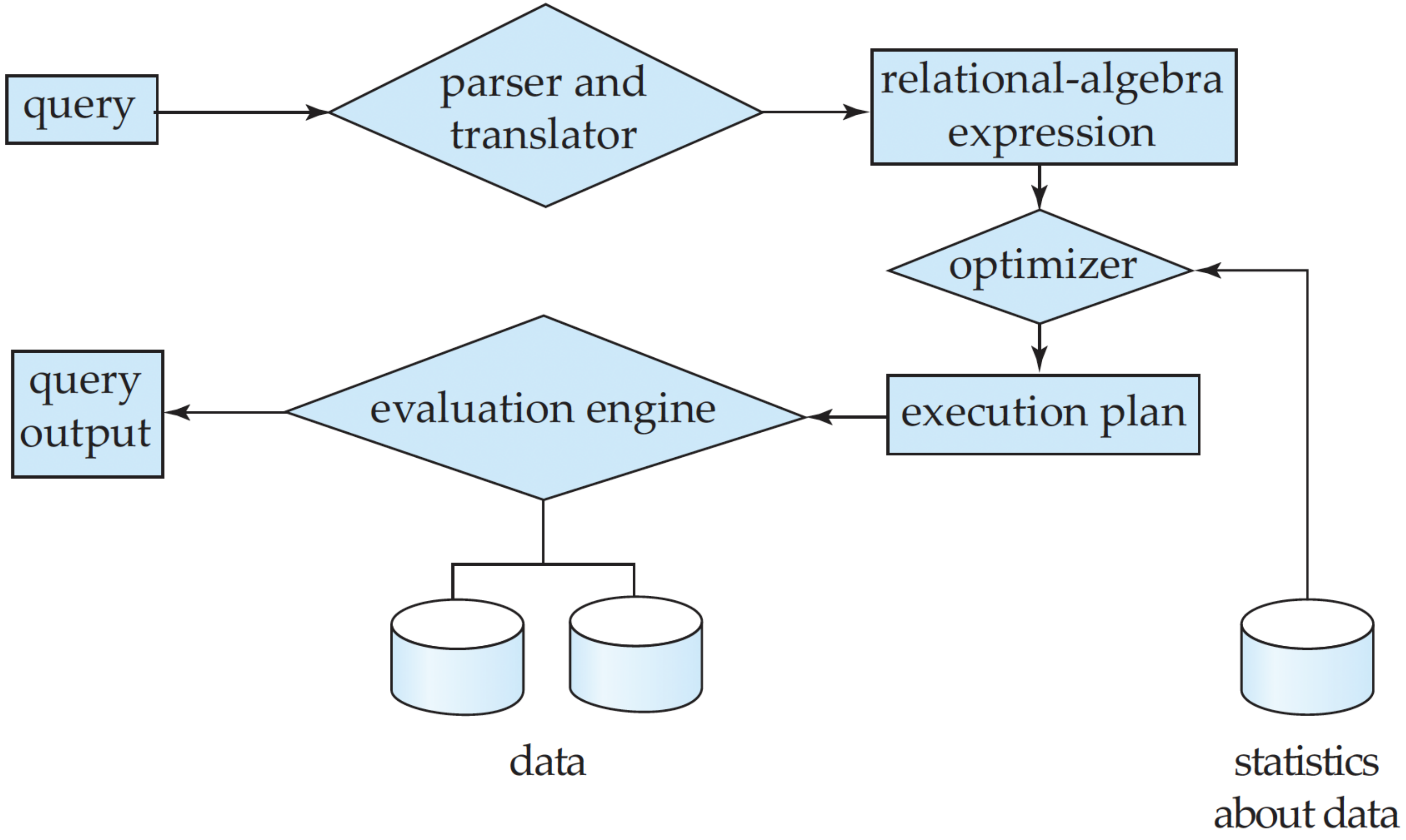


*Diagram from Database Systems Concepts book.

Parser

Parser

- Parsing helps to translate the query into a usable form.

Parser

- Parsing helps to translate the query into a usable form.
- SQL is suitable for human to write queries, but not suited for system's internal representation of a query.

Parser

- Parsing helps to translate the query into a usable form.
- SQL is suitable for human to write queries, but not suited for system’s internal representation of a query.
- Relational algebra is a more suited representation.

Parser

- Parsing helps to translate the query into a usable form.
- SQL is suitable for human to write queries, but not suited for system's internal representation of a query.
- Relational algebra is a more suited representation.
- So, the first step for the system, on receiving a query is to translate it into an understandable format.

Parser

- Does this remind you of some other component that you have used/heard?

Parser

- Does this remind you of some other component that you have used/heard?

- Compilers!
    - A compiler also includes a parser to parse your programs.

Parser

- Does this remind you of some other component that you have used/heard?

- Compilers!
  - A compiler also includes a parser to parse your programs.

- To generate an understandable format, query parser needs to:
  - Check the syntax of the user’s query.
  - Verify that the relation names appearing in the query are names of the relations in the database, and
  - Many more tasks…

- Post this, the system constructs a parse-tree representation of the query.

Parser

- We are not covering parsing as it is part of standard compilers course!

What are the challenges with Parsing a Query

What are the challenges with Parsing a Query

Say, we have the following query:

select salary **from** instructor **where** salary < 75000;

What is the relational algebra translation for this query?

What are the challenges with Parsing a Query

Say, we have the following query:

select salary **from** instructor **where** salary < 75000;

What is the relational algebra translation for this query?

- $\boldsymbol{\sigma}_{\text{salary<7500}}$ ($\boldsymbol{\pi}_{\text{salary}}$ (instructor));

What are the challenges with Parsing a Query

Say, we have the following query:

select salary **from** instructor **where** salary < 75000;

What is the relational algebra translation for this query?

- $\boldsymbol{\sigma}_{salary<7500}$ ($\boldsymbol{\pi}_{salary}$ (instructor));

Can there be another translation?

What are the challenges with Parsing a Query

Say, we have the following query:

select salary **from** instructor **where** salary < 75000;

What is the relational algebra translation for this query?

- $\boldsymbol{\sigma}_{salary<7500}(\boldsymbol{\pi}_{salary}(\text{instructor}))$; **Can there be another translation?**

- $\boldsymbol{\pi}_{salary}(\boldsymbol{\sigma}_{salary<7500}(\text{instructor}))$;

Query Evaluation

- You have the parsed query in a relational algebra format, so what next?

Query Evaluation

- You have the parsed query in a relational algebra format, so what next?
- You need to evaluate the query (compute the result of the query).

What are the challenges with Evaluating a Query

What are the challenges with Evaluating a Query

Say, we selected the following relational algebra translation for the earlir query.

- $\boldsymbol{\sigma}_{\text{salary<7500}}$ ($\boldsymbol{\pi}_{\text{salary}}$ (instructor));

How will you execute this query?

What are the challenges with Evaluating a Query

Say, we selected the following relational algebra translation for the earlir query.

- $\boldsymbol{\sigma}_{\text{salary<7500}}$ ($\boldsymbol{\pi}_{\text{salary}}$ (instructor));

How will you execute this query?

1) Search every tuple in instructor to find tuples with salary less than 75000.

What are the challenges with Evaluating a Query

Say, we selected the following relational algebra translation for the earlir query.

- $\boldsymbol{\sigma}_{\text{salary<7500}}$ ($\boldsymbol{\pi}_{\text{salary}}$ (instructor));

How will you execute this query?

1) Search every tuple in instructor to find tuples with salary less than 75000.

2) Say you have a B^+-tree index on salary, we can use that index to locate the tuples.

Evaluation Plan

- Query translation should specify two things:
    - The corresponding relational-algebra expression
    - Annotate it with instructions stating how to evaluate each operation.

- **Annotations** can state the algorithm to be used, or one or more indices to use.

Evaluation Plan

- Query translation should specify two things:
  - The corresponding relational-algebra expression
  - Annotate it with instructions stating how to evaluate each operation.

- **Annotations** can state the algorithm to be used, or one or more indices to use.

- **Evaluation Primitive**:
  - A relational algebra operation annotated with instructions on how to evaluate it.

- **Query-Execution plan**:
  - A sequence of primitive operations that help to evaluate a query.

A simple query Evaluation Plan

$$\begin{tikzcd} \boldsymbol{\pi}_{\text{salary}} \arrow[d, dash] \\ \boldsymbol{\sigma}_{\text{salary}<7500\ ;} \text{ use index 1} \arrow[d, dash] \\ \text{instructor;} \end{tikzcd}$$

→ Here, index 1 could be an index on salary; internally numbered as 1.

Cost Optimization

- Different evaluation plans for a given query can have different costs.

Cost Optimization

- Different evaluation plans for a given query can have different costs.
- You cannot expect users to write queries in a way that it results in the most efficient evaluation plan.
- Responsibility of the DBMS to construct a query evaluation plan that minimizes the cost of query evaluation.
  - This task is called as **query optimization**.

How to Measure cost of a Query?

- Recall that the cost to access data from disk is the most expensive as disk accesses are slow compared to in-memory operations.

- **Cost of Query Evaluation plan** =
  - Number of block transfers from disk + Number of disk seeks.

- Say;
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - To transfer **b** blocks and perform **s** disk seeks would take = **b*D + s*A**

How to Measure cost of a Query?

- Several other factors also contribute to query cost:
    - **Writes are twice expensive as reads**. Why?

How to Measure cost of a Query?

- Several other factors also contribute to query cost:
  - **Writes are twice expensive as reads**. Why?

    Because disk systems read sectors back after they are written to verify that the write was successful.

How to Measure cost of a Query?

- Several other factors also contribute to query cost:
  - **Writes are twice expensive as reads**. Why?

    Because disk systems read sectors back after they are written to verify that the write was successful.

  - You may have to even estimate the **cost of writing the final result** of an operation back to the disk.

Scans and Searching

- We will now spend time on estimating the cost of search and scans.
- What are the important factors to consider when estimating the cost of a search or scan?

Scans and Searching

- We will now spend time on estimating the cost of search and scans.
- What are the important factors to consider when estimating the cost of a search or scan?
  - Is the search on a primary key?
  - Is the search equal to primary key?
  - Do you have access to indexes on primary key?
  - Do you have access to secondary indexes?

File Scans: Linear Search

- Let’s assume we have a query like the following:

select * **from** instructors;

- This is going to scan over all the records in the table.

- A query similar to scan is where you have to search a key and you do not have any index → You are forced to perform linear search.

- So, what do you think is the cost of this query?

File Scans: Linear Search

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - Say, you have to transfer **b** blocks.
- What is the cost of linear search or scan?

File Scans: Linear Search

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - Say, you have to transfer **b** blocks.

- What is the cost of linear search or scan?
  - **A + b * D**
  - One initial seek (**A**) to reach the correct block/sector.
  - Then, transferring **b** blocks.

Equality Search on Primary Key

- Let's assume we have a query like the following:

select * **from** instructors **where** id = 10;

- The difference between this query and the previous query is that this query wants us to search a specific record.

- Further, assume that the attribute "id" is the primary key.

- And, suppose we have a B^+-tree index built over the attribute "id".

Equality Search on Primary Key

- Let’s assume we have a query like the following:

select * **from** instructors **where** id = 10;

- The difference between this query and the previous query is that this query wants us to search a specific record.

- Further, assume that the attribute “id” is the primary key.

- And, suppose we have a B^+-tree index built over the attribute “id”.

- So, **what do you think is the cost of this query**?

Equality Search on Primary Key

- Let’s assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)

- What is the cost of Equality Search on Primary Key?

Equality Search on Primary Key

- Let's assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)

- What is the cost of Equality Search on Primary Key?
  - **(h+1) * (A + D)**
  - Why this?

Equality Search on Primary Key

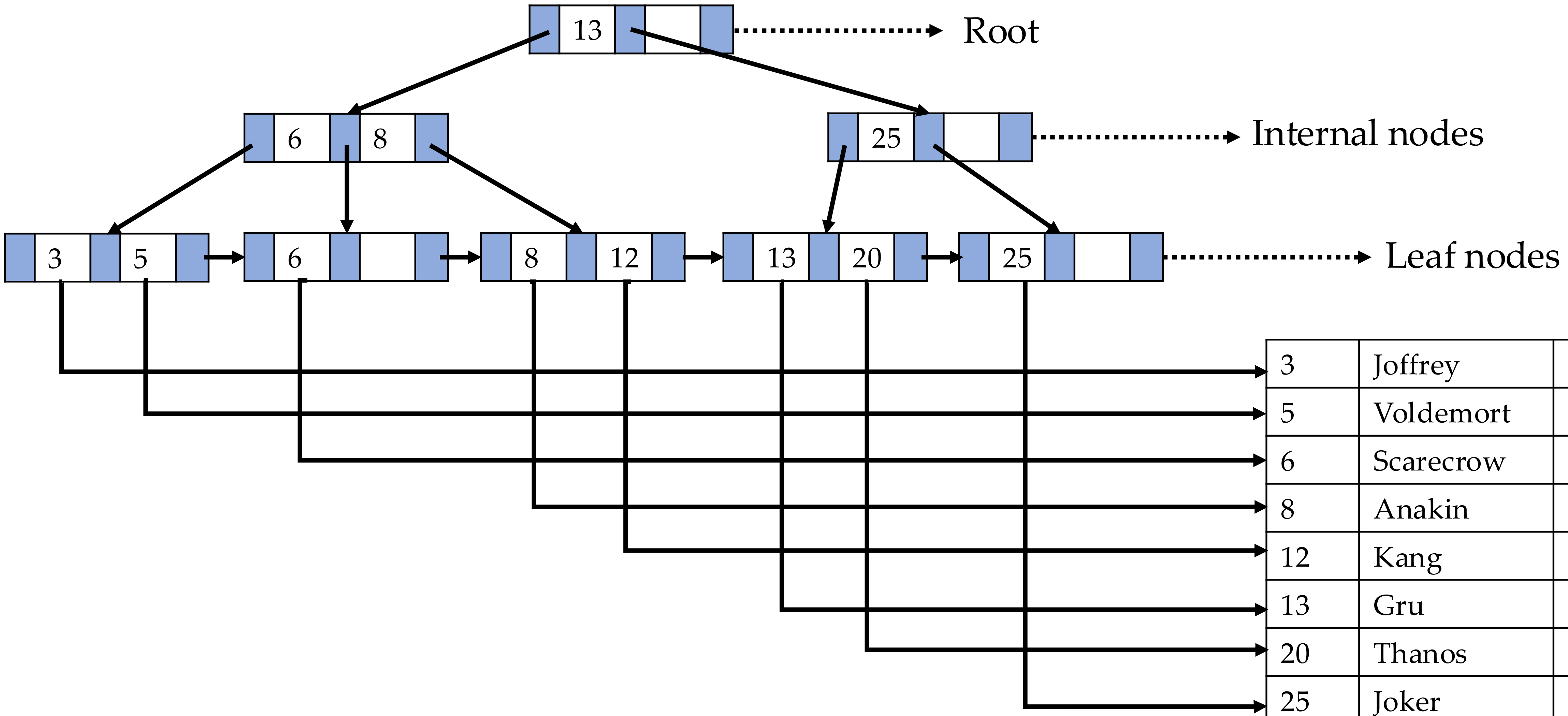
- These are worst-case cost estimates and the assumption is that the B^+-tree is large and cannot be stored in-memory.
  - So, every node in the B^+-tree has to be fetched from the disk.

Equality Search on Primary Key

- These are worst-case cost estimates and the assumption is that the B^+-tree is large and cannot be stored in-memory.
    - So, every node in the B^+-tree has to be fetched from the disk.

- All the keys in a B^+-tree are at the leaf nodes. So,
    - First, you need to traverse all the way to leaf node.
    - During this traversal, you will access B^+-tree nodes to determine which path to take.
    - Fetching each of these nodes is a disk access and requires seek time.
    - And, then once you reach the desired node, you need to also fetch the actual record.

Equality Search on Primary Key



3	Joffrey	18	600
5	Voldemort	70	400
6	Scarecrow	30	150
8	Anakin	20	200
12	Kang	20	500
13	Gru	45	100
20	Thanos	100	400
25	Joker	66	200

File

Equality Search on Non-Key

- Let’s assume we have a query like the following:

select * **from** instructors **where** age = 45;

- The difference between this query and the previous query is that this query wants us to search a specific record on a non-key.

- Here, the attribute “age” is not the primary key.

- So, what is the challenge with estimating the cost of this query?

Equality Search on Non-Key

- B^+-tree index is built over the primary key attribute.
- For non-key attributes, we do not have an available index.
- So, we need to fetch multiple records and blocks as the index is not much useful.

Equality Search on Non-Key

- Let's assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - **b** → Number of blocks to fetch

- What is the cost of Equality Search on Non-Key?

Equality Search on Non-Key

- Let’s assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - **b** → Number of blocks to fetch

- What is the cost of Equality Search on Non-Key?
  - **(h+1) * (A+D) + b * D**

Equality Search on Non-Key

- **(h+1) * (A+D) + b * D**

- Most of the equation looks same, but **Why b*D?**

Equality Search on Non-Key

- **(h+1) * (A+D) + b * D**
- Most of the equation looks same, but **Why b*D?**
- Because, we do not know which block the data that we want resides, so we need to access multiple blocks till we find the one.
- The good thing is as all blocks are stored contiguously, so no extra seek time!

Equality Search on Secondary Key

- Let’s assume we have a query like the following:

select * **from** instructors **where** age = 45;

- But, now assume we have a second index on the attribute “age”.

- Further, assume that our secondary index is also a B^+-tree.

- So, what is cost of this query?

Equality Search on Secondary Key

- Let’s assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)

- What is the cost of Equality Search on Secondary Key?
  - **(h+1) * (A + D)**
  - Why is this same as Equality Search on Primary Key?

Equality Search on Secondary Key

- The cost is same as equality on primary key because we have a B^+-tree and we are doing an exact match!
    - No extra blocks accessed.

Equality Search on Secondary Non-Key

- Let's assume we have a query like the following:

select * **from** instructors **where** salary = 10000;

- But, now assume that we have a
    - Primary index on "Id".
    - Secondary index on "age".
    - But, **no index for "salary"**.

- So, what is cost of this query?

Equality Search on Secondary Non-Key

- Let’s assume the **height of the tree** total levels from root to leaf) = **h.**

- Given,
  - **D** → Time to transfer a block from disk.
  - **A** → Block access time (Seek time + Rotational Latency)
  - **n** → Number of records to fetch

- What is the cost of Equality Search on Non-Key?
  - **(h+n) * (A+D)**
  - Why “n”?

Equality Search on Secondary Non-Key

- Recall that data on the disk is stored sequentially according to the primary index.

- If you are going to access the data according to the primary index, you will incur seek cost only once.

Equality Search on Secondary Non-Key

- Recall that data on the disk is stored sequentially according to the primary index.
- If you are going to access the data according to the primary index, you will incur seek cost only once.
- When you search with the help of a secondary index, you lose the benefit of sequential data access.

Equality Search on Secondary Non-Key

- Recall that data on the disk is stored sequentially according to the primary index.
- If you are going to access the data according to the primary index, you will incur seek cost only once.
- When you search with the help of a secondary index, you lose the benefit of sequential data access.
- As you are searching on a secondary non-key, something for which index does not exist, then you will be fetching random blocks → seek cost.

Comparative Search on Primary Key

- Let's assume we have the following queries:

select * **from** instructors **where** id < 10;

select * **from** instructors **where** id <= 10;

- For these queries, we need to decide whether we want to use a B+-tree index or linear scan.
  - Why do we need to decide?

Comparative Search on Primary Key

- Let's assume we have the following queries:

select * **from** instructors **where** id < 10;

select * **from** instructors **where** id <= 10;

- For these queries, we need to decide whether we want to use a B$^+$-tree index or linear scan.
    - Why do we need to decide?

Comparative Search on Primary Key

- Let's assume we have the following queries:

select * **from** instructors **where** id < 10;

select * **from** instructors **where** id <= 10;

- For these queries, we need to decide whether we want to use a B^+-tree index or linear scan.
    - Why do we need to decide?

- Because linear scan will often result in cheaper estimate as we have to get all the values less or less than equal to the primary key.

Comparative Search on Primary Key

- Let's assume we have the following queries:

select * **from** instructors **where** id > 10;

select * **from** instructors **where** id >= 10;

- For these queries, we will make use of a B+-tree index , followed by a file scan.
  - Why do we need to decide?

Comparative Search on Primary Key

- Let’s assume we have the following queries:

select * **from** instructors **where** id > 10;

select * **from** instructors **where** id >= 10;

- For these queries, we will make use of a B+-tree index, followed by a file scan.
  - Why do we need to decide?

- Because the last record to access is the end of the file.

- What is the cost?

Comparative Search on Primary Key

- Cost same as equality on non-key.
- Let's assume the **height of the tree** total levels from root to leaf) = **h.**
- Given,
    - **D** → Time to transfer a block from disk.
    - **A** → Block access time (Seek time + Rotational Latency)
    - **b** → Number of blocks to fetch
- What is the cost of Equality Search on Non-Key?
    - **(h+1) * (A+D) + b * D**